**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO TIỂU LUẬN**

DỮ LIỆU LỚN

Đề tài:

Phân tích dữ liệu sản phẩm bộ nhớ trên Amazon bằng thuật toán phân cụm

**Giảng viên hướng dẫn:**

ThS. Nguyễn Hồ Duy Trí

**Sinh viên thực hiện:**

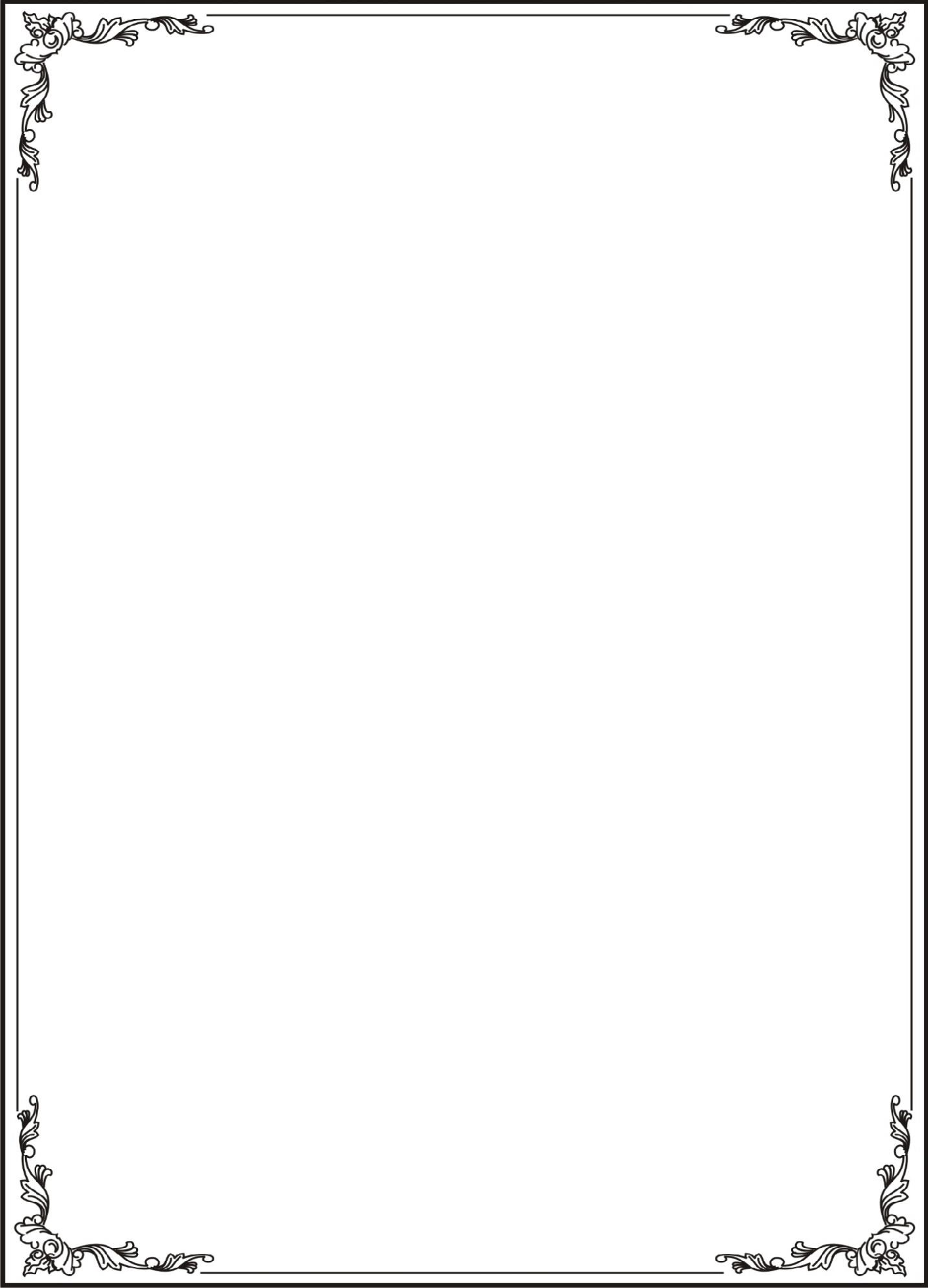
Nguyễn Hoàng Đăng Khoa – 21520999

Cù Ngọc Hoàng - 21522086

Nguyễn Trần Gia Kiệt - 21522258

Bùi Đình Triệu - 21521576

*Thành phố Hồ Chí Minh, tháng 12 năm 2024*

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO TIỂU LUẬN**

DỮ LIỆU LỚN

Đề tài:

Phân tích dữ liệu sản phẩm bộ nhớ trên Amazon bằng thuật toán phân cụm

**Giảng viên hướng dẫn:**

ThS. Nguyễn Hồ Duy Trí

**Sinh viên thực hiện:**

Nguyễn Hoàng Đăng Khoa – 21520999

Cù Ngọc Hoàng - 21522086

Nguyễn Trần Gia Kiệt - 21522258

Bùi Đình Triệu - 21521576

*Thành phố Hồ Chí Minh, tháng 12 năm 2024*

**LỜI CẢM ƠN**

Trước hết, nhóm chúng em xin gửi lời cảm ơn sâu sắc đến tập thể quý thầy cô trường Đại học Công nghệ Thông tin - Đại học Quốc gia TP.HCM và quý thầy cô khoa Hệ thống thông tin đã tạo điều kiện, giúp chúng em học tập và có được những kiến thức cơ bản làm tiền đề giúp chúng em hoàn thành được dự án này.

Đặc biệt, nhóm chúng em xin gửi lời cảm ơn chân thành và sâu sắc tới thầy Nguyễn Hồ Duy Trí (Giảng viên giảng dạy lý thuyết và thực hành môn DỮ LIỆU LỚN – IS405). Nhờ sự hướng dẫn tận tình và chu đáo của thầy, nhóm chúng em đã học hỏi được nhiều kinh nghiệm và hoàn thành thuận lợi, đúng tiến độ cho dự án của mình.

Ngoài ra, chúng em cũng gửi lời cảm ơn đến tập thể lớp IS405.P11 khoảng thời gian qua đã đồng hành cùng nhau. Cảm ơn sự đóng góp của tất cả các bạn cho những buổi học luôn sôi nổi, thú vị và dễ tiếp thu.

Trong quá trình thực hiện tiểu luận, nhóm chúng em luôn giữ một tinh thần cầu tiến, học hỏi và cải thiện từ những sai lầm, tham khảo từ nhiều nguồn tài liệu khác nhau và luôn mong đạt được kết quả nhất có thể. Tuy nhiên, do vốn kiến thức còn hạn chế trong quá trình trau dồi từng ngày, nhóm chúng em không thể tránh được những sai sót, vì vậy chúng em mong rằng quý thầy cô sẽ đưa ra nhận xét một cách chân thành để chúng em học hỏi thêm kinh nghiệm nhằm mục đích phục vụ tốt các dự án khác trong tương lai. Xin chân thành cảm ơn quý thầy cô!

**Nhóm thực hiện**

**NHẬN XÉT CỦA GIẢNG VIÊN**

*……., ngày……...tháng……năm 2024*

**Người nhận xét**

*(Ký tên và ghi rõ họ tên****)***

**MỤC LỤC**

[CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI 12](#_Toc185471210)

[1.1. Lý do chọn đề tài 12](#_Toc185471211)

[1.2. Giới thiệu nguồn dữ liệu 12](#_Toc185471212)

[1.3. Mô tả dữ liệu 13](#_Toc185471213)

[1.4. Mô tả bài toán 15](#_Toc185471214)

[CHƯƠNG 2: LẤY DỮ LIỆU TỪ AMAZON 17](#_Toc185471215)

[2.1. Apache Airflow 17](#_Toc185471216)

[2.1.1. Giới thiệu 17](#_Toc185471217)

[2.1.2. Kiến trúc 17](#_Toc185471218)

[2.2. Crawl data từ Amazon phục vụ cho đồ án 18](#_Toc185471219)

[2.2.1. Sử dụng Docker để setup Airflow 18](#_Toc185471220)

[2.2.2. Crawl Data từ Amazon Error! Bookmark not defined.](#_Toc185471221)

[CHƯƠNG 3: KỸ THUẬT TIỀN XỬ LÝ 26](#_Toc185471222)

[3.1. Làm sạch dữ liệu 26](#_Toc185471223)

[3.1.1. Loại bỏ cột bị thiếu dữ liệu 26](#_Toc185471224)

[3.1.2. Xử lý cột price, old\_price, rating 26](#_Toc185471225)

[3.1.3. Xử lý cột reviews, purchases 28](#_Toc185471226)

[3.1.4. Xử lý cột Brands 30](#_Toc185471227)

[3.1.5. Xử lý cột Memory Storage Capacity, Digital Storage Capacity 32](#_Toc185471228)

[3.1.6. Xử lý cột Hardware Interface 35](#_Toc185471229)

[3.1.7. Xử lý cột Write Speed, Read Speed 37](#_Toc185471230)

[3.1.8. Xử lý cột Color 39](#_Toc185471231)

[3.1.9. Loại bỏ các hàng bị thiếu dữ liệu và bị lặp 42](#_Toc185471232)

[3.2. Chuẩn bị dữ liệu cho kỹ thuật khai thác 42](#_Toc185471233)

[3.2.1. Kỹ thuật MinMaxScaler: 42](#_Toc185471234)

[3.2.2. Kỹ thuật StringIndex + One-hot Encoding: 43](#_Toc185471235)

[CHƯƠNG 4: KỸ THUẬT KHAI THÁC DỮ LIỆU 45](#_Toc185471236)

[4.1. K-Means 45](#_Toc185471237)

[4.1.1. Cơ sở lý thuyết 45](#_Toc185471238)

[4.1.2. Thực nghiệm 47](#_Toc185471239)

[4.2. DBScan 53](#_Toc185471240)

[4.2.1. Cơ sở lý thuyết 53](#_Toc185471241)

[4.2.2. Thực nghiệm 58](#_Toc185471242)

[CHƯƠNG 5: KẾT QUẢ ĐẠT ĐƯỢC 66](#_Toc185471243)

[5.1. Kết quả 66](#_Toc185471244)

[5.1.1. K-Means 66](#_Toc185471245)

[5.1.2. DBScan 67](#_Toc185471246)

[5.2. So sánh, đánh giá 68](#_Toc185471247)

[5.2.1. K-Means 68](#_Toc185471248)

[5.2.2. DBScan 72](#_Toc185471249)

[CHƯƠNG 6: KẾT LUẬN 75](#_Toc185471250)

[6.1. Ưu điểm 75](#_Toc185471251)

[6.2. Hạn chế 75](#_Toc185471252)

[6.3. Hướng phát triển 75](#_Toc185471253)

[PHÂN CÔNG CÔNG VIỆC 77](#_Toc185471254)

[TÀI LIỆU THAM KHẢO 79](#_Toc185471255)

**DANH MỤC HÌNH ẢNH**

[Figure 1. Trang Amazon 13](#_Toc185502285)

[Figure 2. File Excel dữ liệu 13](#_Toc185502286)

[Figure 3. Hiển thị dưới dạng Dataframe 15](#_Toc185502287)

[Figure 4. Biểu đồ phần trăm dữ liệu bị thiếu mỗi cột 15](#_Toc185502288)

[Figure 5 Xây dựng Image từ Docker File 18](#_Toc185502289)

[Figure 6 Cấu hình Docker Compose 19](#_Toc185502290)

[Figure 7 Hàm giả lập User-Agent để tránh bị phát hiện là bot 19](#_Toc185502291)

[Figure 8 Hàm Random thời gian chống bot 20](#_Toc185502292)

[Figure 9 Hàm lấy tiêu đề sản phẩm 20](#_Toc185502293)

[Figure 10 Hàm lấy giá hiện tại của sản phẩm 20](#_Toc185502294)

[Figure 11 Hàm lấy giá gốc của sản phẩm 20](#_Toc185502295)

[Figure 12 Lấy URL sản phẩm 21](#_Toc185502296)

[Figure 13 Lấy đánh giá sản phẩm 21](#_Toc185502297)

[Figure 14 Hàm lấy số lượng đánh giá 21](#_Toc185502298)

[Figure 15 Hàm số lượng bán ra 21](#_Toc185502299)

[Figure 16 Một phần của hàm lấy thông tin chi tiết sản phẩm 22](#_Toc185502300)

[Figure 17 Một phần của hàm xử lí các tác vụ chính 22](#_Toc185502301)

[Figure 18 Các tham số DAG 23](#_Toc185502302)

[Figure 19 Các tham số gọi hàm thực hiện 23](#_Toc185502303)

[Figure 20 Chia nhiều file thực hiện 24](#_Toc185502304)

[Figure 21 Kết quả file CSV 24](#_Toc185502305)

[Figure 22 Kết quả file json 24](#_Toc185502306)

[Figure 23 Hàm merge file csv (1) 25](#_Toc185502307)

[Figure 24 Hàm merge file csv (2) 25](#_Toc185502308)

[Figure 25. Các cột có thể được giữ lại 26](#_Toc185502309)

[Figure 26. Đổi tên cột 26](#_Toc185502310)

[Figure 27. Xử ký cột price, old\_price và rating 27](#_Toc185502311)

[Figure 28. Kết quả price, old\_price và rating 27](#_Toc185502312)

[Figure 29. Xử lý cột reviews, purchases 28](#_Toc185502313)

[Figure 30. Kết quả reviews, purchases 29](#_Toc185502314)

[Figure 31. Đếm số Brands 30](#_Toc185502315)

[Figure 32. Lọc Brands 31](#_Toc185502316)

[Figure 33. Xử lý cột Brands 31](#_Toc185502317)

[Figure 34. Kết quả Brands 32](#_Toc185502318)

[Figure 35. Xử lý cột Memory Storage Capacity, Digital Storage Capacity (1) 33](#_Toc185502319)

[Figure 36. Xử lý cột Memory Storage Capacity, Digital Storage Capacity (2) 33](#_Toc185502320)

[Figure 37. Xử lý cột Memory Storage Capacity, Digital Storage Capacity (3) 34](#_Toc185502321)

[Figure 38. Xử lý cột Memory Storage Capacity, Digital Storage Capacity (4) 34](#_Toc185502322)

[Figure 39. Kết quả Memory Storage Capacity, Digital Storage Capacity 35](#_Toc185502323)

[Figure 40. Xử lý cột Hardware Interface 36](#_Toc185502324)

[Figure 41. Kết quả Hardware Interface 37](#_Toc185502325)

[Figure 42. Xử lý cột Write Speed 38](#_Toc185502326)

[Figure 43. Kết quả Write Speed 38](#_Toc185502327)

[Figure 44. Xử lý cột Read Speed 39](#_Toc185502328)

[Figure 45. Kết quả Read Speed 39](#_Toc185502329)

[Figure 46. Đếm số Color 40](#_Toc185502330)

[Figure 47. Xử lý cột Color 41](#_Toc185502331)

[Figure 48. Kết quả Color 41](#_Toc185502332)

[Figure 49. Loại bỏ hàng bị lặp và bị thiếu dữ liệu 42](#_Toc185502333)

[Figure 50. Kết quả dataframe sau làm sạch dữ liệu 42](#_Toc185502334)

[Figure 51. Kỹ thuật MinMaxScaler lên các cột dữ liệu số 43](#_Toc185502335)

[Figure 52. StringIndexer + One-hot Encoding 44](#_Toc185502336)

[Figure 53 Điều kiện dừng 47](#_Toc185502337)

[Figure 54. Eps-neighborhood 54](#_Toc185502338)

[Figure 55. Directly density-reachable 55](#_Toc185502339)

[Figure 56. Density-reachable 56](#_Toc185502340)

[Figure 57. Density-connected 56](#_Toc185502341)

[Figure 58. Phân loại các điểm trong DBScan 57](#_Toc185502342)

[Figure 59. Mô tả song song hóa giải thuật DBScan 60](#_Toc185502343)

[Figure 60. Xử lý dữ liệu đầu vào 61](#_Toc185502344)

[Figure 61. Tính khoảng cach Euclidean giữa các điểm 61](#_Toc185502345)

[Figure 62. Khởi tạo eps và minPts và broadcast giá trị 62](#_Toc185502346)

[Figure 63. Tìm hàng xóm cho tất cả các điểm 62](#_Toc185502347)

[Figure 64. Phân loại điểm core và khởi tạo cluster ban đầu cho các điểm 62](#_Toc185502348)

[Figure 65. Tìm các điểm noise 63](#_Toc185502349)

[Figure 66. Tìm các điểm core 63](#_Toc185502350)

[Figure 67. Tìm các điểm border 63](#_Toc185502351)

[Figure 68. Cập nhật các điểm border vào dataframe 63](#_Toc185502352)

[Figure 69. Mở rộng cụm từ một điểm core 64](#_Toc185502353)

[Figure 70. Hàm mở rộng cụm 65](#_Toc185502354)

[Figure 71 Số lượng dữ liệu ở mỗi cụm 66](#_Toc185502355)

[Figure 72. Kết quả phân cụm bằng DBScan 67](#_Toc185502356)

[Figure 73 Chỉ số silhouette score cài đặt không dùng thư viện máy học 69](#_Toc185502357)

[Figure 74 Chỉ số silhouette score cài đặt dùng thư viện máy học 70](#_Toc185502358)

[Figure 75 Phân cụm không dùng thư viện máy học 71](#_Toc185502359)

[Figure 76 Phân cụm sử dụng thư viện máy học 72](#_Toc185502360)

[Figure 77. Chỉ số Silhoutte của DBScan 73](#_Toc185502361)

[Figure 78. Heatmap của DBScan 73](#_Toc185502362)

# TỔNG QUAN ĐỀ TÀI

## Lý do chọn đề tài

* Trong thời đại thương mại điện tử bùng nổ, Amazon đã trở thành một trong những nền tảng mua sắm trực tuyến lớn nhất toàn cầu với hàng triệu sản phẩm thuộc nhiều danh mục khác nhau. Điều này đặt ra thách thức lớn cho cả người tiêu dùng và doanh nghiệp trong việc lựa chọn sản phẩm, tối ưu hóa chiến lược kinh doanh, và phân tích dữ liệu trên quy mô lớn.
* Việc phân cụm các loại sản phẩm trên Amazon giúp cung cấp một cái nhìn tổng quan về cơ cấu nhóm sản phẩm và mang lại nhiều lợi ích thiết thực như:
* Cá nhân hóa trải nghiệm người dùng.
* Hỗ trợ doanh nghiệp tối ưu chiến lược marketing
* Quản lý kho hàng hiệu quả.
* Khám phá xu hướng thị trường và nhóm sản phẩm tiềm năng.
* Với những lợi ích trên, nhóm quyết định sử dụng thuật toán phân cụm K-Means và DBSCAN để phân loại các sản phẩm bộ nhớ dựa trên bộ dữ liệu thu thập từ Amazon, phục vụ mục tiêu học tập và nghiên cứu.

## Giới thiệu nguồn dữ liệu

* Nguồn dữ liệu: <https://www.amazon.com/s?i=computers-intl-ship&bbn=16225007011&rh=n%3A16225007011%2Cn%3A1292110011>

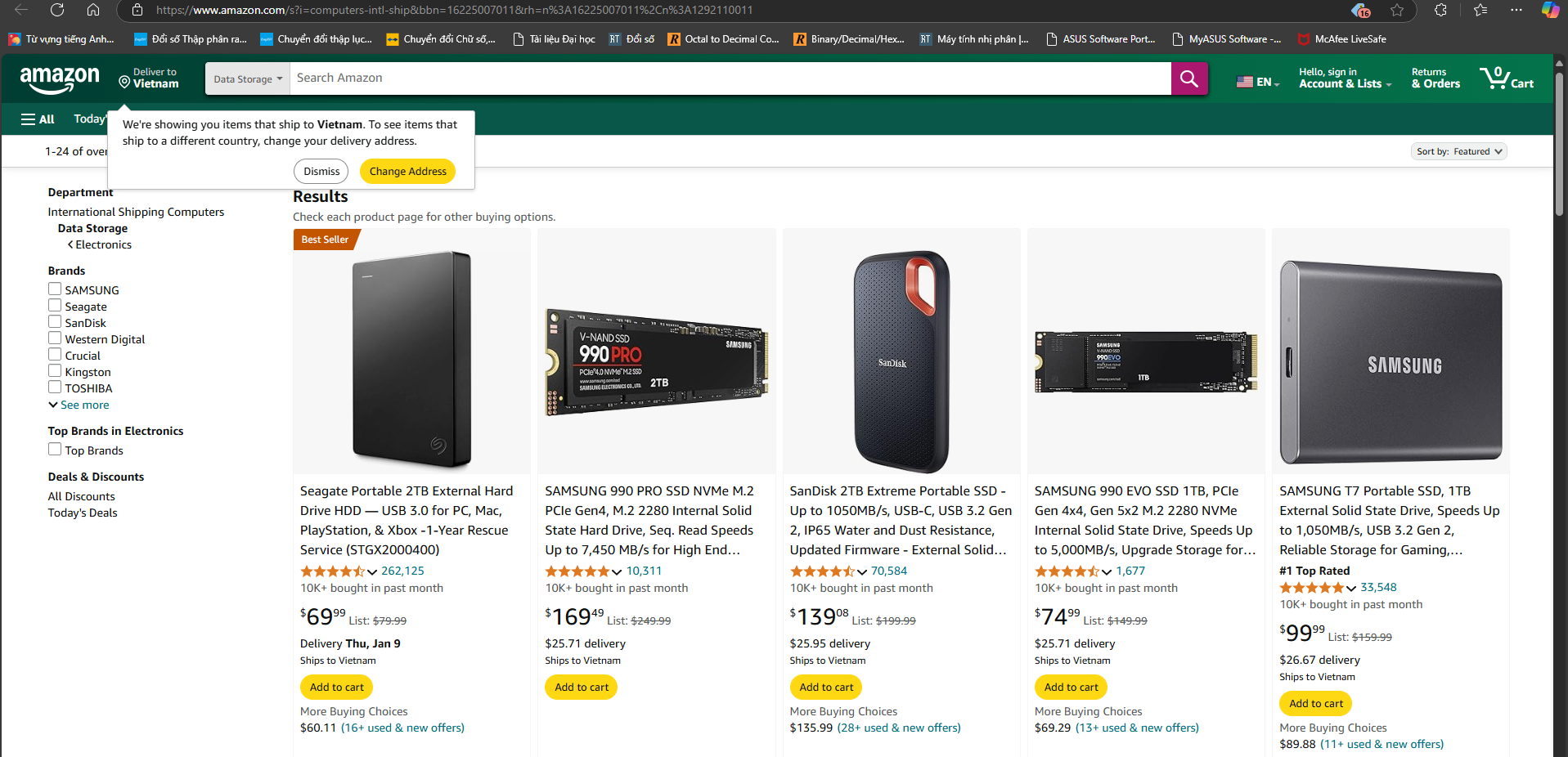


Figure 1. Trang Amazon

* Nguồn dữ liệu được thu nhập từ trang amazon về các sản phẩm bộ nhớ bao gồm có 3740 hàng và 140 cột dữ liệu.
* Mỗi hàng dữ liệu sẽ bao gồm thông tin của sản phẩm bộ nhớ, được chia thành 2 loại chủ yếu là USB và SSD + HDD. Các thông tin bao gồm, giá cả, số điểm đánh giá, lượt đánh giá, thông số kỹ thuật, etc...

## Mô tả dữ liệu

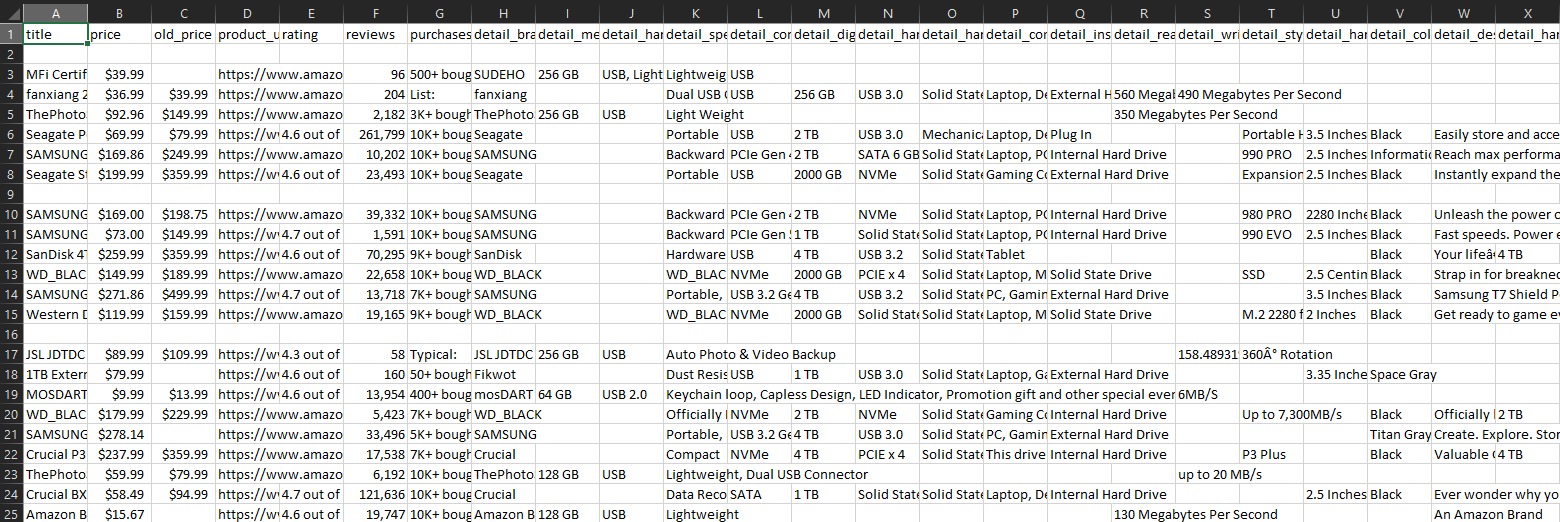


Figure 2. File Excel dữ liệu

|  |  |
| --- | --- |
| Thuộc tính | Mô tả |
| title | Tiêu đề sản phẩm |
| price | Giá hiện tại của sản phẩm |
| old\_price | Giá gốc của sản phẩm (nếu có) |
| product\_url | Liên kết đến trang chi tiết sản phẩm |
| rating | Đánh giá trung bình của sản phẩm |
| reviews | Số lượng đánh giá |
| purchases | Số lượng đã mua |
| detail\_brand | Thương hiệu sản phẩm |
| detail\_memory\_storage\_capacity | Dung lượng lưu trữ |
| detail\_hardware\_interface | Giao diện phần cứng |
| detail\_special\_feature | Tính năng đặc biệt |
| detail\_read\_speed | Tốc độ đọc dữ liệu |
| detail\_write\_speed | Tốc độ ghi dữ liệu |
| detail\_item\_dimensions | Kích thước sản phẩm |
| detail\_weight | Trọng lượng sản phẩm |
| detail\_operating\_system | Hệ điều hành tương thích |
| detail\_specific\_uses\_for\_product | Mục đích sử dụng cụ thể |
| detail\_color | Màu sắc của sản phẩm |
| detail\_model\_name | Tên model của sản phẩm |
| detail\_material | Vật liệu của sản phẩm |
| detail\_included\_components | Các thành phần đi kèm |
| detail\_compatible\_devices | Các thiết bị tương thích |
| detail\_data\_transfer\_rate | Tốc độ truyền dữ liệu |
| detail\_power\_source | Nguồn năng lượng |

**Đánh giá sơ bộ:**

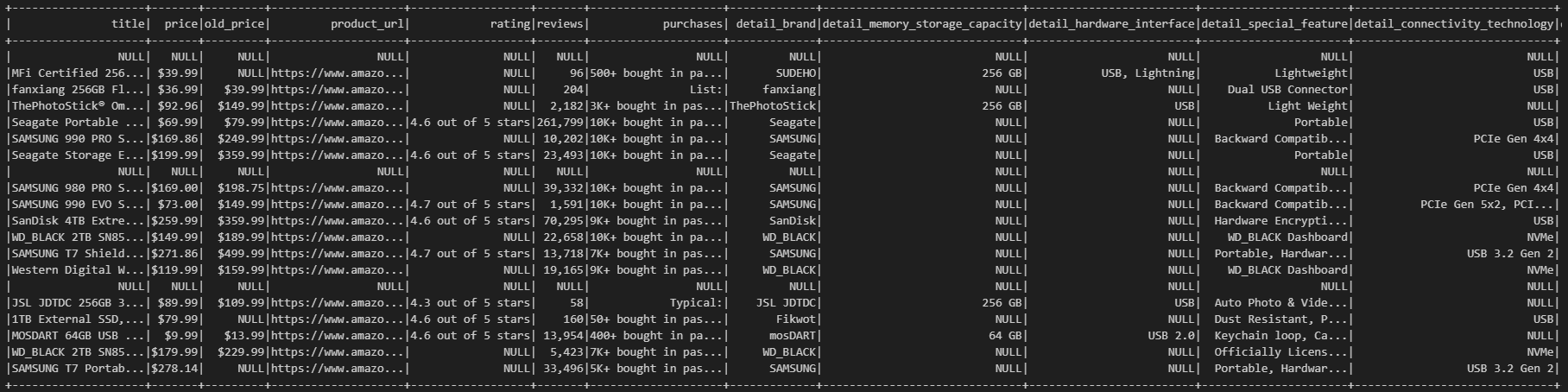


Figure 3. Hiển thị dưới dạng Dataframe

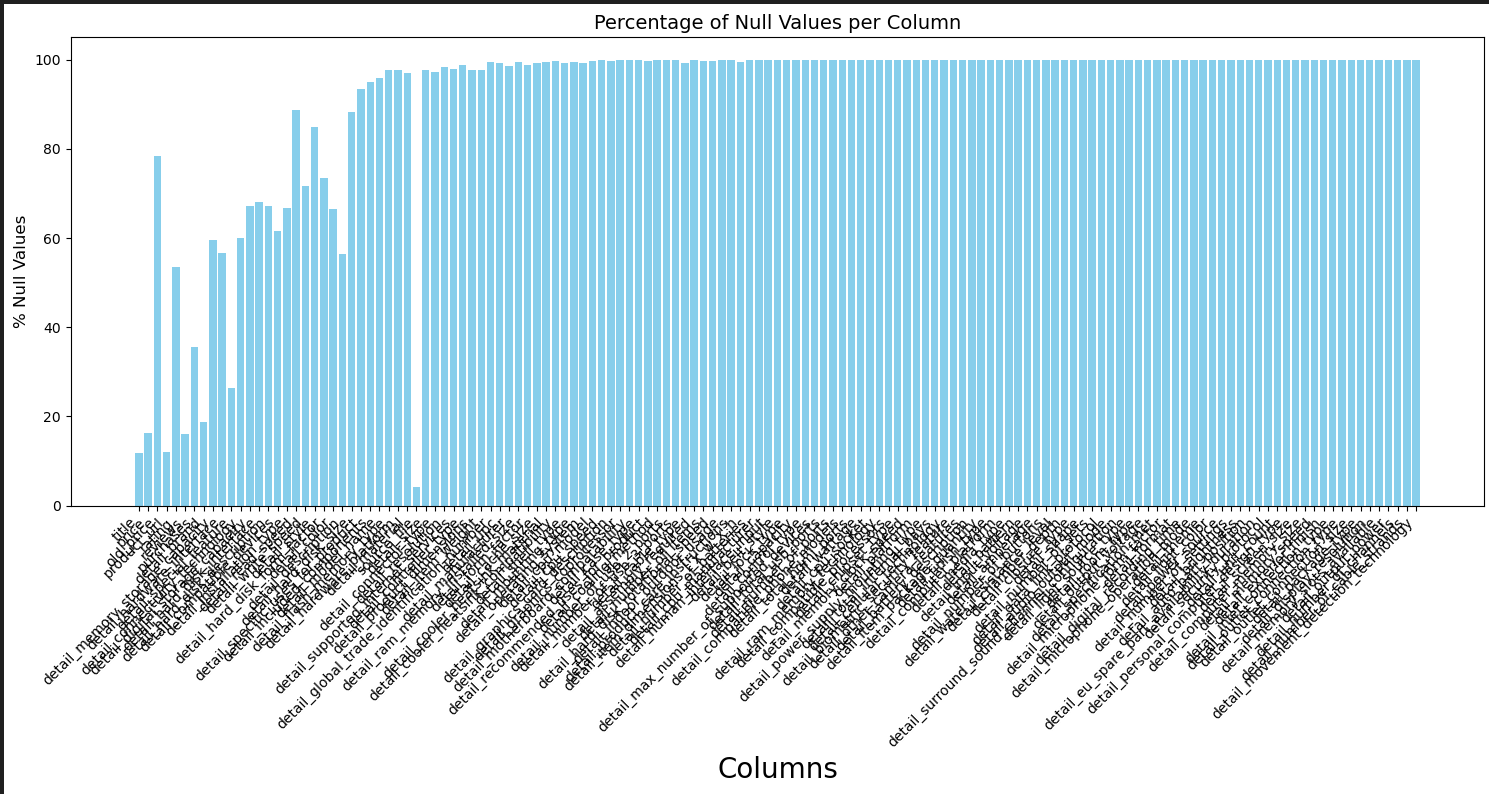


Figure 4. Biểu đồ phần trăm dữ liệu bị thiếu mỗi cột

* Biểu đồ trên cho ta thấy lượng lớn dữ liệu bị thiếu, nhiều cột gần như không có dữ liệu.
* Cột title qua đánh giá sơ bộ có thể chứa dữ liệu liên quan đến thông số của sản phẩm, có thể khai thác trong quá trình làm sạch dữ liệu để có thể dữ lại một só cột.
* Do dữ liệu bị thiếu quá nhiều nên chỉ có thể tập trung xử lý trước vấn đề bị thiểu do không có có dữ liệu đủ để khai thác.

## Mô tả bài toán

* Trong báo cáo này, nhóm thực hiện phương pháp phân cụm bằng thuật toán KMeans và DBScan dựa trên sự tương đồng của dữ liệu và tìm ra tập luật để dự đoán. Từ đó giúp rõ hơn các sản phầm bộ nhớ được phân phối và bán trên sàn thương mại điện tử Amazon.

# LẤY DỮ LIỆU TỪ AMAZON

## Apache Airflow

### Giới thiệu

* Apache Airflow là một nền tảng mã nguồn mở dùng để phát triển, lên lịch và giám sát các luồng công việc theo lô (batch-oriented workflows). Khung mở rộng bằng Python của Airflow cho phép bạn xây dựng các luồng công việc kết nối với hầu như bất kỳ công nghệ nào. Giao diện web hỗ trợ quản lý trạng thái của các luồng công việc. Airflow có thể triển khai theo nhiều cách, từ một tiến trình duy nhất trên máy tính cá nhân của bạn đến một cấu hình phân tán để hỗ trợ ngay cả các luồng công việc lớn nhất.

### Kiến trúc

* Kiến trúc của Airflow bao gồm nhiều thành phần. Các phần sau đây mô tả chức năng của từng thành phần và liệu chúng có cần thiết cho việc cài đặt Airflow ở mức tối thiểu hay thành phần tùy chọn để đạt được khả năng mở rộng, hiệu suất và khả năng mở rộng Airflow tốt hơn
* **Các thành phần cần thiết trong Airflow:** Một cài đặt tối thiểu của Airflow cần bao gồm các thành phần sau:

#### **Scheduler**

**Nhiệm vụ**:

Lên lịch và kích hoạt các luồng công việc (workflows) theo thời gian định trước.

Gửi các task đến executor để thực thi.

**Executor**:

Là một thuộc tính cấu hình của Scheduler, không phải là một thành phần tách biệt.

Chạy trong cùng tiến trình với Scheduler.

Airflow cung cấp nhiều loại executor có sẵn, hoặc bạn có thể tự viết executor của riêng mình.

#### **Webserver**

Cung cấp giao diện người dùng để kiểm tra, kích hoạt và gỡ lỗi DAGs (Directed Acyclic Graphs) và các task.

#### **Thư mục DAG Files**

Chứa các file định nghĩa DAG.

Scheduler đọc các file này để xác định task cần chạy và thời điểm thực thi.

#### **Metadata Database**

Lưu trữ trạng thái và thông tin lịch sử của workflows và tasks.

Cơ sở dữ liệu này là thành phần bắt buộc để Airflow hoạt động.

Hướng dẫn cài đặt Metadata Database có thể được tham khảo trong tài liệu Set up a Database Backend.

## Crawl data từ Amazon phục vụ cho đồ án

### Sử dụng Docker để setup Airflow

* Chọn phiên bản Airflow phù hợp với python để sử dụng trong Docker

A screen shot of a computer

Description automatically generated

Figure 5 Xây dựng Image từ Docker File

* Sử dụng docker compose để chạy container làm server cho airflow

A screenshot of a computer

Description automatically generated

Figure 6 Cấu hình Docker Compose

### Crawl Data từ Amazon

* Tạo tiêu đề HTTP (headers) ngẫu nhiên, bao gồm các User-Agent khác nhau, nhằm giả lập các trình duyệt khác nhau khi gửi request.

A computer screen shot of a program

Description automatically generated

Figure 7 Hàm giả lập User-Agent để tránh bị phát hiện là bot

* Tạo một khoảng thời gian trễ ngẫu nhiên từ 5 đến 7 giây để tránh bị phát hiện là bot khi gửi nhiều request đến Amazon.

A black background with white text

Description automatically generated

Figure 8 Hàm Random thời gian chống bot

* Lấy tiêu đề sản phẩm từ HTML sử dụng BeautifulSoup. Tìm các phần tử phù hợp dựa trên class CSS.

A computer screen shot of a code

Description automatically generated

Figure 9 Hàm lấy tiêu đề sản phẩm

* Lấy giá hiện tại của sản phẩm.

A computer screen shot of text

Description automatically generated

Figure 10 Hàm lấy giá hiện tại của sản phẩm

* Lấy giá gốc của sản phẩm nếu có giảm giá.



Figure 11 Hàm lấy giá gốc của sản phẩm

* Tạo URL đầy đủ dẫn đến sản phẩm dựa trên thuộc tính href của thẻ HTML.



Figure 12 Lấy URL sản phẩm

* Lấy đánh giá của sản phẩm



Figure 13 Lấy đánh giá sản phẩm

* Lấy số lượng đánh giá cho sản phẩm.

A screen shot of a computer

Description automatically generated

Figure 14 Hàm lấy số lượng đánh giá

* Lấy số lượng mua hàng nếu có thông tin này.

A black background with white text

Description automatically generated

Figure 15 Hàm số lượng bán ra

* Lấy thêm thông tin chi tiết từ trang sản phẩm như mô tả, thương hiệu, kiểu dáng (style), và các đặc điểm kỹ thuật khác.

A screen shot of a computer program

Description automatically generated

Figure 16 Một phần của hàm lấy thông tin chi tiết sản phẩm

* Hàm chính thực hiện các tác vụ: khởi tạo thư mục để lưu trữ dữ liệu, lặp qua các trang, thu thập dữ liệu sản phẩm bằng các hàm, ghi dữ liệu thành các file json và csv.

A screenshot of a computer

Description automatically generated

Figure 17 Một phần của hàm xử lí các tác vụ chính

* Định nghĩa các tham số mặc định cho DAG

A computer screen shot of a program code

Description automatically generated

Figure 18 Các tham số DAG

* Tên dag, mô tả, lịch trình, nhiệm vụ gọi hàm chính thực hiện thu thập dữ liệu

A computer screen with text on it

Description automatically generated

Figure 19 Các tham số gọi hàm thực hiện

* Chia thành nhiều file để thực hiện crawl cùng lúc nhiều danh sách trang

A screenshot of a computer

Description automatically generated

Figure 20 Chia nhiều file thực hiện

* Kết quả File CSV và Json

A screen shot of a computer screen

Description automatically generated

Figure 21 Kết quả file CSV

A screen shot of a computer

Description automatically generated

Figure 22 Kết quả file json

* Merge file CSV sau khi thu thập dữ liệu

A screen shot of a computer program

Description automatically generated

Figure 23 Hàm merge file csv (1)

A screen shot of a computer program

Description automatically generated

Figure 24 Hàm merge file csv (2)

# KỸ THUẬT TIỀN XỬ LÝ

## Làm sạch dữ liệu

### Loại bỏ cột bị thiếu dữ liệu

* Do dữ liệu bị thiếu nhiều, chỉ có thể giữ lại một số cột đầu có thể khai thác được thông tin từ cột title.

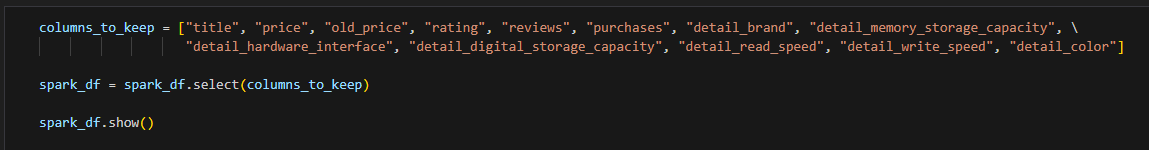


Figure 25. Các cột có thể được giữ lại

* Thông qua đánh giá title, các cột trên là các cột có thể khai thác tốt dữ liệu từ cột title. Sau đó ta sẽ đổi tên một số cột được giữ lại để thuận tiện trong quá trình tiền xử lý.

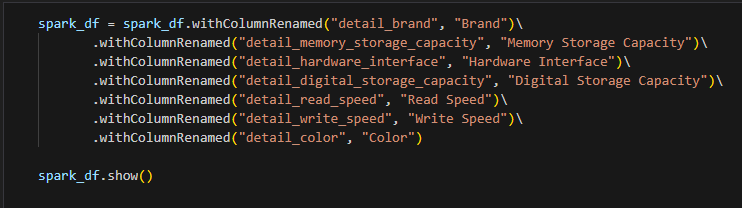


Figure 26. Đổi tên cột

### Xử lý cột price, old\_price, rating

* Đối với hai cột price và old\_price, ta cần loại bỏ ký hiệu ‘$’ để có thể chuyển dữ liệu về lại dạng double (hoặc double) để cho thuận tiến công đoạn tính toánt trong lúc khai thác dữ liệu.
* Trong lúc xử lý, nếu cột old\_price bị thiếu dữ liêu, ta sẽ lấy giá trị từ cột price để điền vào, trong trường hợp price cũng bị thiếu thì ta sẽ bỏ qua và loại hàng dữ liệu đó sau pha làm sạch dữ liệu.
* Đối với cột rating thì ta cần biến kiểu dữ liệu dạng string về lại định dạng double, nếu dữ liệu không có thì ta sẽ điền giá trị 0.

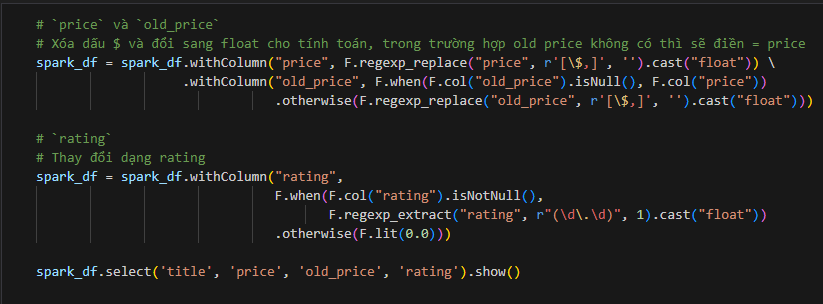


Figure 27. Xử ký cột price, old\_price và rating

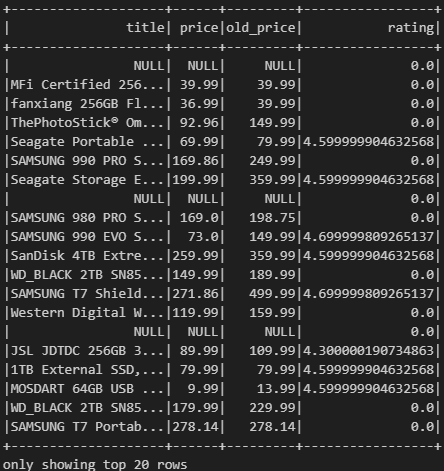


Figure 28. Kết quả price, old\_price và rating

### Xử lý cột reviews, purchases

* Đối với cột reviews, ta chỉ cần đổi lại dữ liệu thành dạng số (int).
* Đối với cột purchases, kiểu dữ liệu ban đầu là string, số lượng sản phẩm bán ra được ở tháng trước (VD: 10K+ bought in last month). Cần phải lấy giá trị ra và chuyển đổi về dạng số.

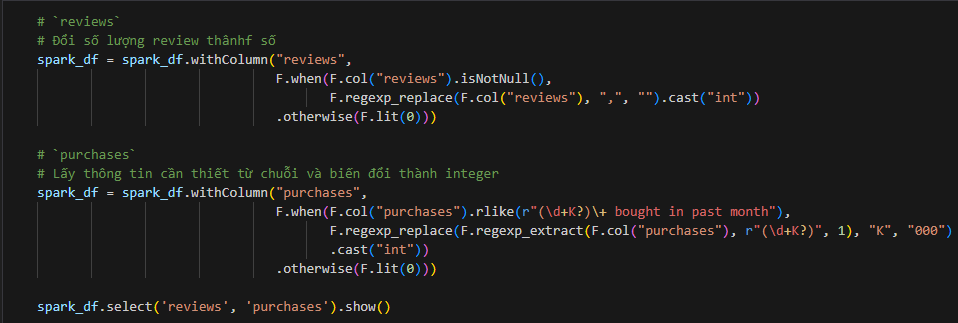


Figure 29. Xử lý cột reviews, purchases

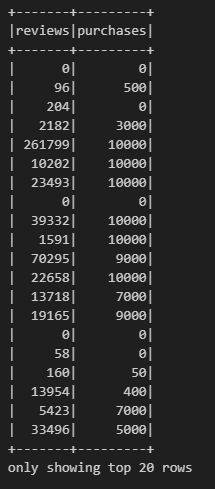


Figure 30. Kết quả reviews, purchases

### Xử lý cột Brands

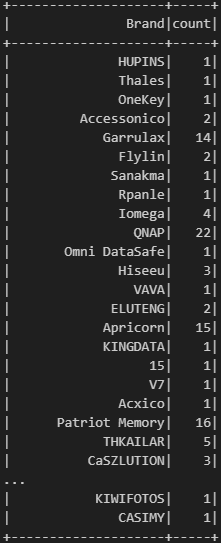


Figure 31. Đếm số Brands

* Dựa trên hình, ta thấy cột Brand nhiều hảng lớn nhỏ khác nhau, điều này có thể dẫn tới nhiễu dữ liệu do quá nhiều giá trị nên trước tiên, ta sẽ lọc bớt các hãng nhỏ lẽ chỉ xuất hiện vài lần với số lần xuất hiện phải ít nhất trên 10 lần mới được chấp nhận (Threshold > 10).

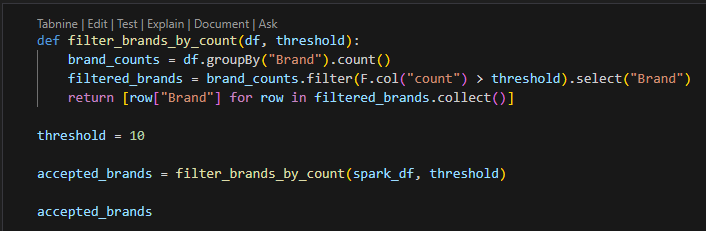


Figure 32. Lọc Brands

* Mảng accepted\_brands chứa các hãng xuất hiện trên 10 lần, từ đây, nếu giá trị là trống thì trước tiên sẽ kiếm dữ liệu trong title và so sánh với mảng accepted\_brands, nếu phù hợp sẽ điền vào giá trị còn nếu không thì sẽ điền ‘others’ đại diện cho các hãng nhỏ hoặc không có. Trong trường hợp dữ liệu đã có thì sẽ so sánh với mảng, nếu không hợp sẽ tương tự điền vào ‘others’.
* Sau khi hoàn thành bước trên thì ta sẽ chuẩn hóa cột theo lowercase.

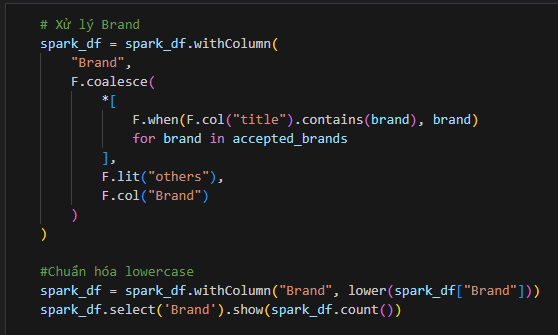


Figure 33. Xử lý cột Brands

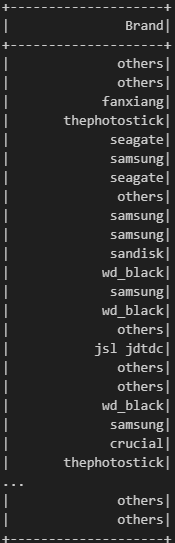


Figure 34. Kết quả Brands

### Xử lý cột Memory Storage Capacity, Digital Storage Capacity

* Do dữ liệu của sản phẩm bộ nhớ được chia thành hai loại là USB và SSD, HDD, nên tương tự ta sẽ có Memory Storage Capacity chứa thông tin về độ lớn của USB và Digital Storage Capacity.
* Tương tự như chiến thuật xử lý đề ra ban đầu, ta sẽ tìm dữ liệu trong title trong trường hợp bị thiếu, nhưng trước tiên cần so sánh xem sản phẩm đó thuôc loại nào.
* Sau khi toàn bộ dữ liệu được điền vào, ta sẽ chuẩn hóa toàn bộ về đơn vị GB. Ngoài ra, nếu sản phẩm là có dữ liệu một bên và không có bên còn lại, giá trị 0 sẽ được điền vào chỗ trống.

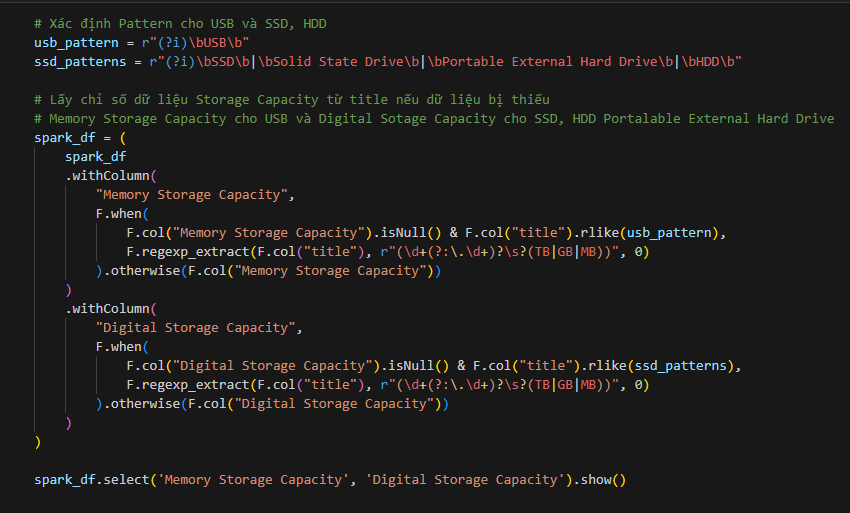


Figure 35. Xử lý cột Memory Storage Capacity, Digital Storage Capacity (1)

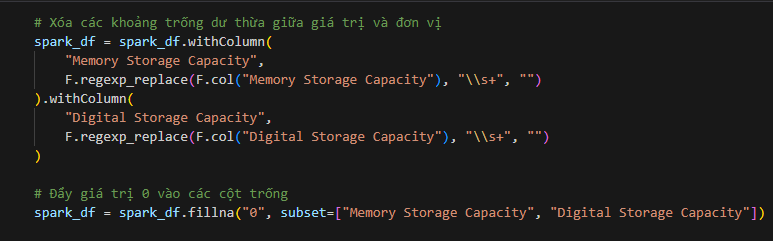


Figure 36. Xử lý cột Memory Storage Capacity, Digital Storage Capacity (2)

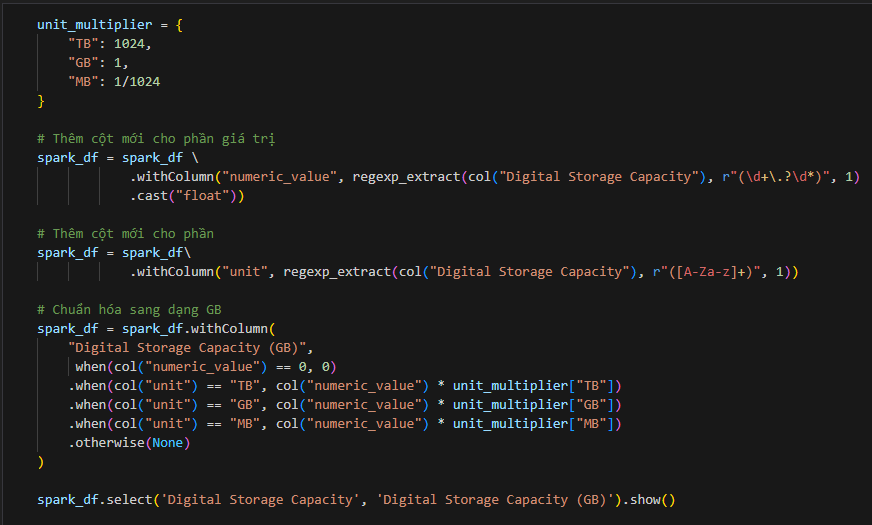


Figure 37. Xử lý cột Memory Storage Capacity, Digital Storage Capacity (3)

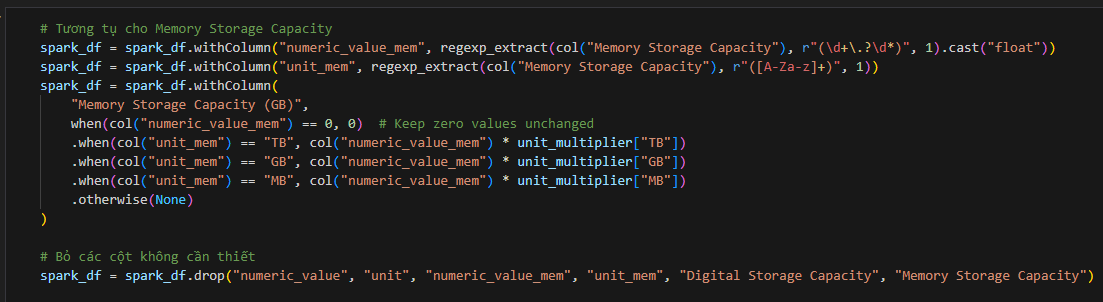


Figure 38. Xử lý cột Memory Storage Capacity, Digital Storage Capacity (4)

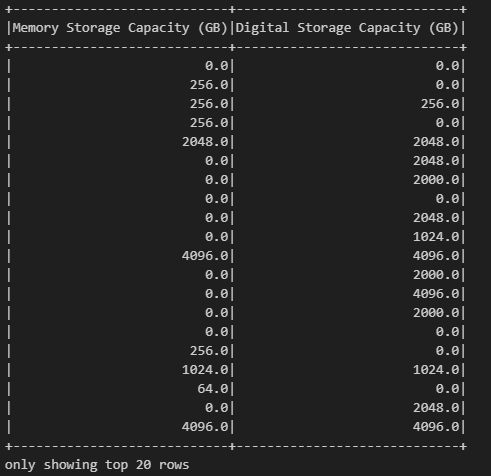


Figure 39. Kết quả Memory Storage Capacity, Digital Storage Capacity

### Xử lý cột Hardware Interface

* Tương tự cột Brand nhưng không cần phải lọc ra các Interface do số lượng Interface nhỏ.

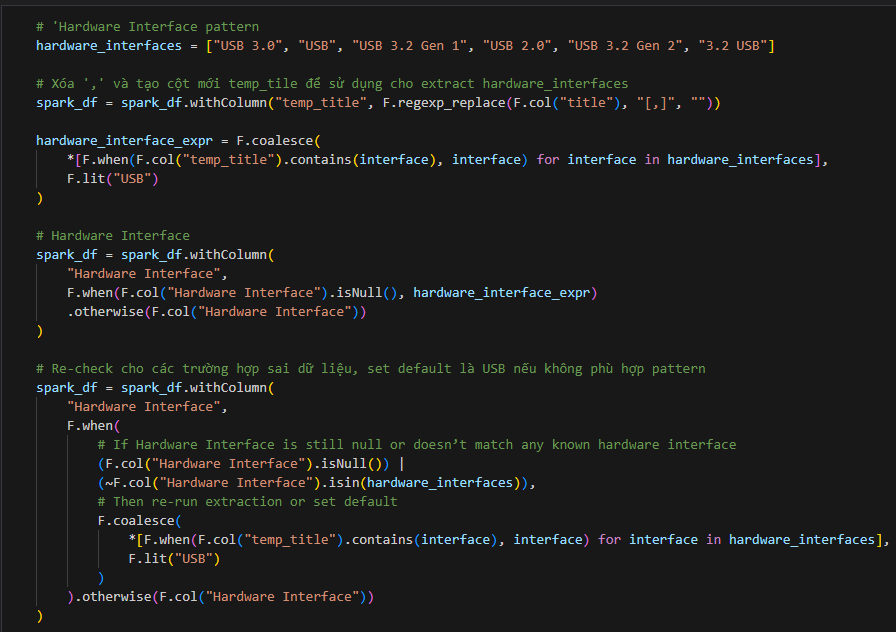


Figure 40. Xử lý cột Hardware Interface

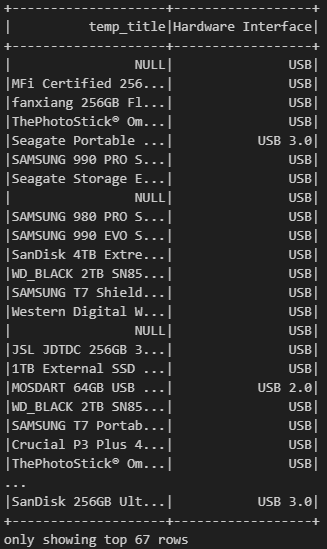


Figure 41. Kết quả Hardware Interface

### Xử lý cột Write Speed, Read Speed

* Tương tự như cột Memory Storage Capacity, ta sẽ tìm dữ liệu trong title nếu dữ liệu trống và bỏ qua nếu dữ liệu đã có. Trong trường hợp nếu title không có đữ liệu thì sẽ điền 0.
* Đối với Read Speed, tương tự như Write Speed nhưng trong trường hợp dữ liệu bị thiếu thì ta sẽ lấy dữ liệu của Write Speed điền vào Read Speed.

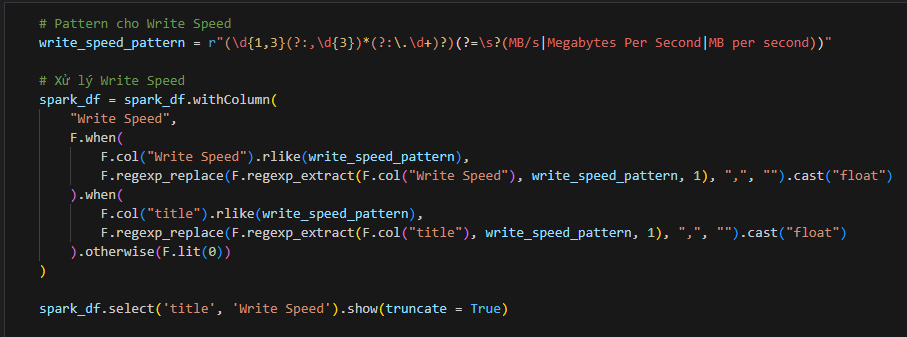


Figure 42. Xử lý cột Write Speed

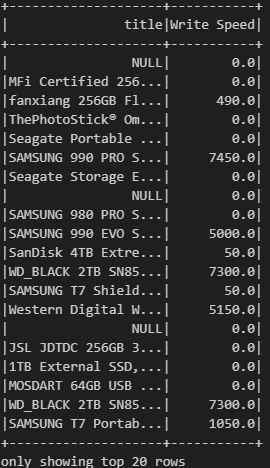


Figure 43. Kết quả Write Speed

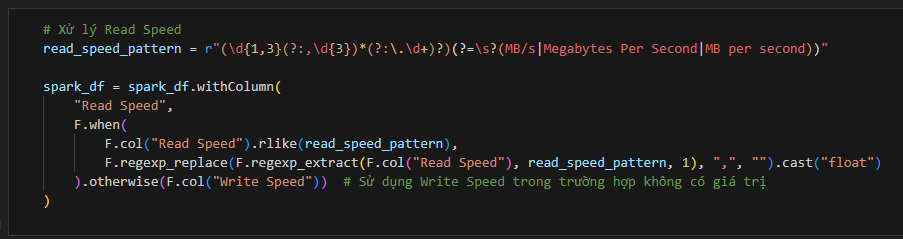


Figure 44. Xử lý cột Read Speed

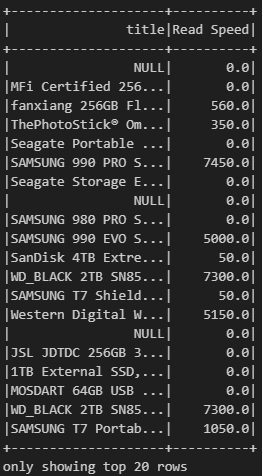


Figure 45. Kết quả Read Speed

### Xử lý cột Color

* Tương tự như Brands, đánh giá sơ bộ cho ta thấy không những xuất hiện lượng lớn Color khác nhau, cột cũng bị nhiễu dữ liệu do các dữ liệu không liên quan xuất hiện.

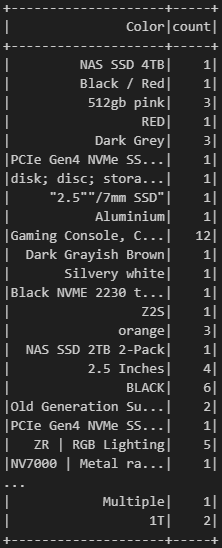


Figure 46. Đếm số Color

* Bắt buộc phải tạo hẵn một mảng chứa các màu định sẵn để xử lý.

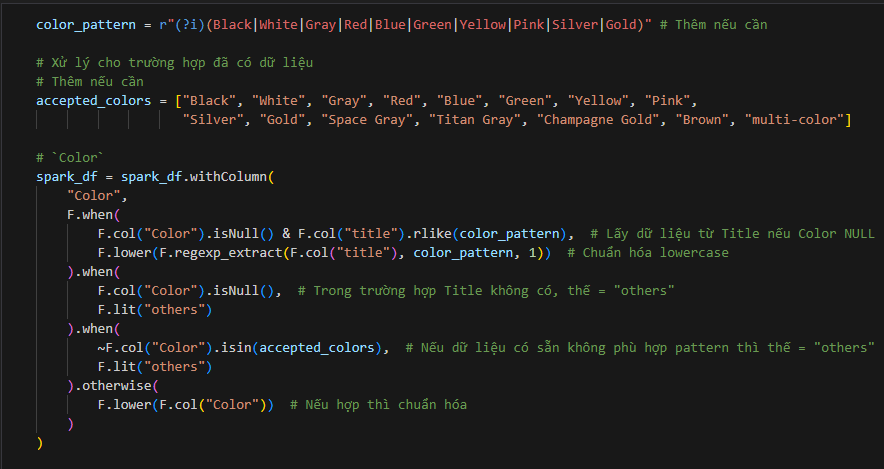


Figure 47. Xử lý cột Color



Figure 48. Kết quả Color

### Loại bỏ các hàng bị thiếu dữ liệu và bị lặp

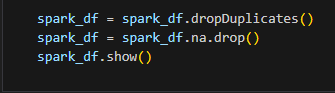


Figure 49. Loại bỏ hàng bị lặp và bị thiếu dữ liệu

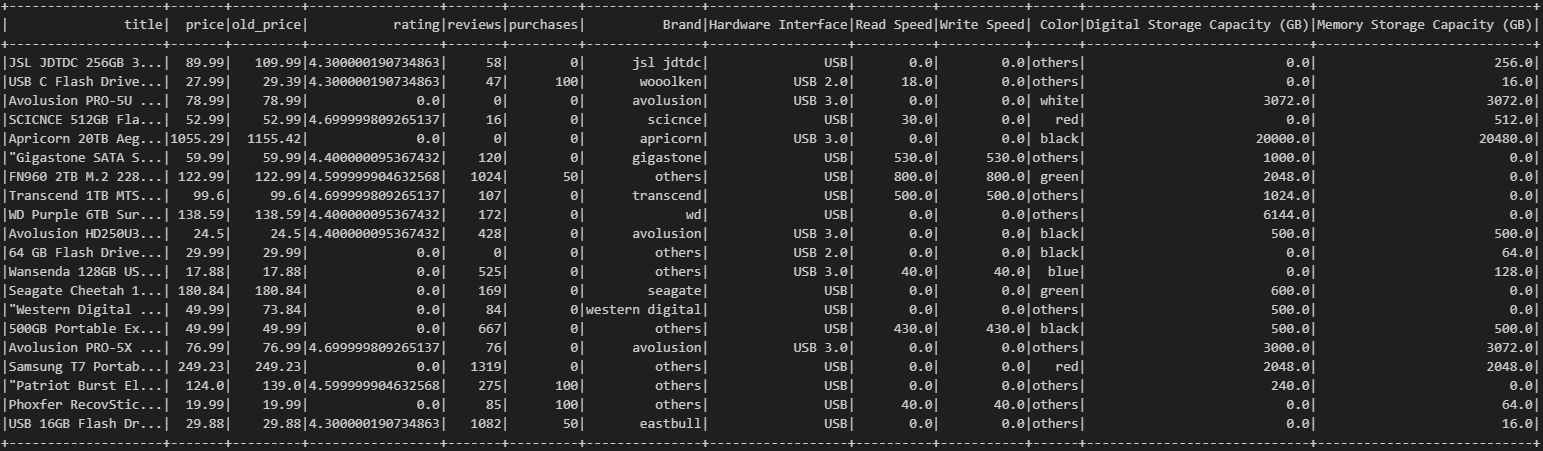


Figure 50. Kết quả dataframe sau làm sạch dữ liệu

## Chuẩn bị dữ liệu cho kỹ thuật khai thác

### Kỹ thuật MinMaxScaler:

* Min/Max scaling là kỹ thuật lấy từng giá trị trừ đi cho giá trị tối thiểu sau đó chia cho hiệu giá trị lớn nhất và nhỏ nhất.
* Kỹ thuật này chia tỷ lệ lại một đặc tính hoặc giá trị quan sát với giá trị phân phối từ 0 đến 1.



Figure 51. Kỹ thuật MinMaxScaler lên các cột dữ liệu số

### Kỹ thuật StringIndex + One-hot Encoding:

* String Indexer là một kỹ thuật được sử dụng để chuyển đổi các giá trị chuỗi (categorial data) thành giá trị số (numeric data) bằng cách ánh xạ từng giá trị chuỗi trong cột thành một giá trị số nguyên. Ví dụ: ["apple", "banana", "orange"] -> ["0", "1", "2"]. Quá trình này sắp xếp các chuỗi dựa trên tần suất xuất hiện của chúng (giá trị có tần suất cao nhất sẽ được ánh xạ vào số nhỏ nhất, thường là 0).
* One-hot encoding là kỹ thuật chuyển đổi dữ liệu phân loại thành dạng vector nhị phân. Với một cột có 𝑛 giá trị khác nhau (categories), One-hot encoding tạo ra 𝑛 cột mới, trong đó mỗi cột biểu diễn một giá trị duy nhất và chứa giá trị 0 hoặc 1.

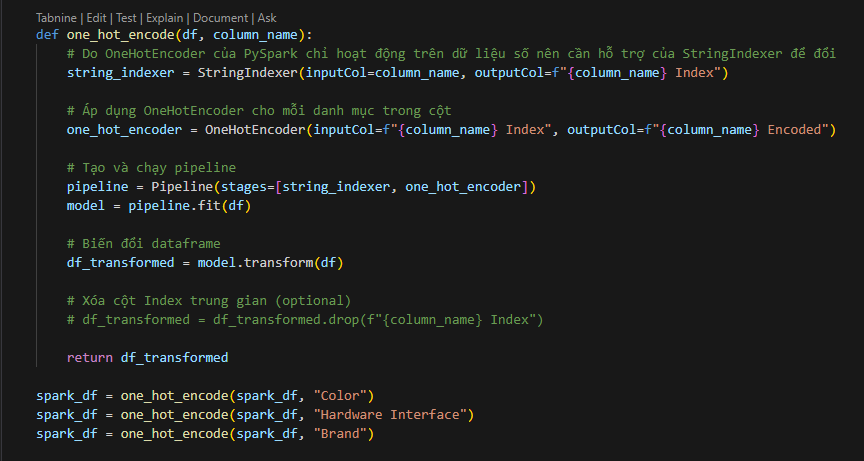
****

Figure 52. StringIndexer + One-hot Encoding

# KỸ THUẬT KHAI THÁC DỮ LIỆU

## K-Means

### Cơ sở lý thuyết

* Thuật ngữ " K -means" được James MacQueen sử dụng lần đầu tiên vào năm 1967, mặc dù ý tưởng này quay trở lại Hugo Steinhaus vào năm 1956. Thuật toán tiêu chuẩn được đề xuất lần đầu tiên bởi Stuart Lloyd của Bell Labs vào năm 1957 như một kỹ thuật cho điều chế mã xung, mặc dù nó không được xuất bản dưới dạng một bài báo cho đến năm 1982. Năm 1965, Edward W. Forgy đã công bố về cơ bản cùng một phương pháp, đó là lý do tại sao nó đôi khi được gọi là Lloyd-Forgy.
* K-Means gán các điểm dữ liệu vào một trong K cụm (clusters) dựa trên khoảng cách của chúng đến tâm cụm (**centroid**). Thuật toán bắt đầu bằng cách chọn ngẫu nhiên các tâm cụm trong không gian dữ liệu. Sau đó, mỗi điểm dữ liệu được gán vào một cụm dựa trên khoảng cách của nó tới tâm cụm gần nhất. Sau khi tất cả điểm dữ liệu được gán cụm, tâm cụm mới được tính lại. Quá trình này lặp đi lặp lại cho đến khi tìm được các cụm tối ưu.
* Trong phân tích này, giả định rằng số lượng cụm K đã được xác định trước và chúng ta cần phân loại các điểm dữ liệu vào một trong các nhóm đó.
* Trong một số trường hợp, **K** không được xác định rõ ràng, và chúng ta cần xác định số lượng cụm K tối ưu. K-means hoạt động tốt nhất khi dữ liệu được phân tách rõ ràng. Khi các điểm dữ liệu chồng lấn nhau, K-means không phải là phương pháp phù hợp. Thuật toán này nhanh hơn so với các kỹ thuật gom cụm khác và tạo sự liên kết chặt chẽ giữa các điểm dữ liệu.
* Tuy nhiên, K-means có một số hạn chế:
  + Không cung cấp thông tin rõ ràng về chất lượng cụm.
  + Kết quả cụm phụ thuộc vào cách khởi tạo tâm cụm ban đầu, có thể dẫn đến các kết quả khác nhau.
  + Thuật toán nhạy cảm với nhiễu (**noise**) và có thể bị kẹt ở cực tiểu cục bộ (**local minima**).
* Mục tiêu của thuật toán K-Means: Mục tiêu của việc gom cụm (**clustering**) là chia tập dữ liệu hoặc quần thể thành các nhóm sao cho các điểm dữ liệu trong cùng một nhóm có tính tương đồng cao hơn so với các điểm dữ liệu trong các nhóm khác. Về cơ bản, đây là một quá trình phân nhóm dựa trên độ tương tự và khác biệt giữa các điểm dữ liệu.
* K-Means hoạt động như thế nào? Giả sử chúng ta có một tập dữ liệu gồm các đối tượng, mỗi đối tượng có các đặc trưng (features) với các giá trị tương ứng (dạng vector). Nhiệm vụ của chúng ta là phân loại các đối tượng này vào các nhóm cụ thể. Để làm điều này, chúng ta sử dụng thuật toán **K-means**, một thuật toán học máy không giám sát. Chữ **K** trong tên thuật toán đại diện cho số lượng nhóm/cụm mà chúng ta muốn phân loại các đối tượng vào
* Giải thuật K-means được thực hiện như sau
  + Khởi tạo các trung tâm cụm ban đầu
    - k là số lượng cụm được xác định trước.
  + Phân cụm dữ liệu
    - Với mỗi điểm dữ liệu, ta sẽ tính khoảng cách của nó tới các trung tâm (bằng Khoảng cách Euclid). Ta sẽ gán chúng vào trung tâm gần nhất. Tập hợp các điểm được gán vào cùng 1 trung tâm sẽ tạo thành cụm.
      * Trong đó
* d(x,y): Khoảng cách giữa hai điểm x và y.
* n: Số chiều (số thành phần) của không gian.
* xi và yi​: Thành phần thứ i của điểm x và y.
  + Cập nhật trung tâm cụm
    - Sau khi gán các điểm dữ liệu vào các cụm, trung tâm cụm m(i) mới sẽ được tính lại bằng trung bình cộng của tất cả các điểm dữ liệu thuộc cụm i:
      * Trong đó
  + Kiểm tra điều kiện dừng: Thuật toán dừng khi các trung tâm cụm ổn định, tức là sự thay đổi giữa các trung tâm nhỏ hơn một ngưỡng nhất định.A screen shot of a computer

    Description automatically generated

Figure 53 Điều kiện dừng

### Thực nghiệm

def calculate\_distance(point1, point2):

    return sum((x - y) \*\* 2 for x, y in zip(point1, point2))

* Hàm calcualate\_distance(point1, point2) dùng để tính khoảng cách Euclidean giữa hai điểm dữ liệu point1 và point2:
  + Dùng công thức
  + Trong code, công thức này được tính bằng cách lặp qua các cặp tọa độ tương ứng (x, y) của hai điểm và tính tổng bình phương hiệu của chúng.

class K\_Mean:

    def \_\_init\_\_(self, k=2, predictCol='prediction', seed=1):

        self.k = k

        self.centroids = None

        self.predictCol = predictCol

        self.seed = seed

* Khởi tạo class K-Means
* \_\_init\_\_: Hàm khởi tạo với các tham số:
  + k: số cụm cần phân
  + predictCol: Tên cột sẽ chứa giá trị dự đoán cụm (mặc định là 'prediction')
  + seed: Dùng để đảm bảo tính ngẫu nhiên có thể tái lập (random seed)
  + centroids: Lưu trữ danh sách tọa độ của các tâm cụm (**centroid**)

def find\_closest\_cluster(self, data\_point):

        distances = [calculate\_distance(data\_point, centroid) for centroid in self.centroids]

        return distances.index(min(distances))

* Hàm find\_closest\_cluster nhằm xác định cụm gần nhất cho một điểm dữ liệu (data\_point)
* Cách thực hiện:
  + Tính khoảng cách từ điểm dữ liệu đến từng tâm cụm.
  + Trả về chỉ số của cụm có khoảng cách nhỏ nhất.

def fit(self, dataframe, feature\_columns):

        df = dataframe.withColumn('features', F.array(feature\_columns))

* Hàm này thực hiện quá trình huấn luyện (training) mô hình K-Means Clustering trên dữ liệu. Dưới đây là giải thích chi tiết từng dòng:
* dataframe: Tập dữ liệu đầu vào dưới dạng DataFrame của PySpark.
* feature\_columns: Danh sách các cột đặc trưng cần sử dụng để gom cụm.
* withColumn('features', F.array(...)): Tạo cột mới tên là features, chứa mảng (array) các giá trị của các cột đặc trưng trong feature\_columns.

for \_ in range(20):

            # Randomly select initial centroids

            self.centroids = df.select('features').rdd.map(lambda r: r.features).takeSample(False, num=self.k, seed=self.seed)

self.seed += 1

* **Lặp 20 lần** để đảm bảo việc chọn tâm cụm ban đầu có chất lượng tốt hơn:
* df.select('features'): Lấy cột features từ DataFrame.
* rdd.map(lambda r: r.features): Chuyển đổi DataFrame thành RDD, chỉ lấy các giá trị của cột features.
* takeSample(False, num=self.k, seed=self.seed): Lấy ngẫu nhiên k điểm từ tập dữ liệu làm tâm cụm ban đầu, sử dụng seed để tạo tính ngẫu nhiên tái lập.
* self.seed += 1: Tăng giá trị seed sau mỗi lần khởi tạo để tránh việc lặp lại cùng một mẫu tâm cụm.

while True:

                # Create a user-defined function to assign clusters

                assign\_cluster\_udf = F.udf(self.find\_closest\_cluster, IntegerType())

* **Vòng lặp chính**: Chạy đến khi các tâm cụm hội tụ (không còn thay đổi đáng kể).
* assign\_cluster\_udf: Tạo một **UDF** (hàm do người dùng định nghĩa), sử dụng hàm find\_closest\_cluster để gán mỗi điểm dữ liệu vào cụm gần nhất.

                # Assign each point to the nearest cluster

                df = df.withColumn( self.predictCol, assign\_cluster\_udf(F.col('features')))

* withColumn(self.predictCol, ...): Thêm một cột mới (self.predictCol, mặc định là prediction) vào DataFrame.
* assign\_cluster\_udf(F.col('features')): Tính toán cụm gần nhất cho mỗi điểm dựa trên vector features.

# Recalculate cluster centroids

                cluster\_averages = (

                    df.rdd

                    .map(lambda row: (row[self.predictCol], row['features']))

                    .groupByKey()

                    .mapValues(lambda features: [

                        sum(dimension) / len(features)

                        for dimension in zip(\*features)

                    ])

                )

* df.rdd: Chuyển đổi DataFrame thành RDD để thao tác theo kiểu phân tán, hỗ trợ xử lý linh hoạt hơn.
* map(lambda row: (row[self.predictCol], row['features'])):
  + **Khóa**: row[self.predictCol] - chỉ số của cụm mà điểm dữ liệu được gán vào (ví dụ: cụm 0, 1, 2...).
  + **Giá trị:** row['features'] - vector đặc trưng của điểm dữ liệu.
* groupByKey(): Nhóm các điểm dữ liệu lại theo cụm:
  + Các điểm có cùng giá trị self.predictCol (tức cùng cụm) được nhóm lại với nhau.
  + Kết quả là một RDD với **khóa là số cụm** và **giá trị là danh sách các vector đặc trưng trong cụm đó**:
* mapValues(...): **Tính tâm cụm mới (centroid)**:
  + features: Danh sách các vector đặc trưng trong mỗi cụm.
  + zip(\*features): Ghép các giá trị của từng chiều đặc trưng lại.
* sum(dimension) / len(features): Tính giá trị trung bình của từng chiều

# Convert cluster averages to list

                new\_centroids = self.centroids[:]

                for cluster, avg\_features in cluster\_averages.collect():

                    new\_centroids[cluster] = avg\_features

* new\_centroids: Sao chép danh sách các tâm cụm hiện tại, để tránh thay đổi trực tiếp dữ liệu gốc.
* for cluster, avg\_features in cluster\_averages.collect(): Duyệt qua từng cụm Với mỗi cặp (cluster, avg\_features) trong danh sách thu thập được. Ở đây collect() chỉ thu thập kết quả tính toán trung bình cụm (không phải dữ liệu gốc), đồng thời số lượng cụm (k) thường nhỏ, nên dữ liệu thu thập đủ nhỏ để xử lý.
* new\_centroids[cluster] = avg\_features: Cập nhật tâm cụm với giá trị trung bình vừa tính.

# Check if centroids have stabilized

                max\_centroid\_shift = max(

                    calculate\_distance(self.centroids[i], new\_centroids[i]) for i in range(self.k)

                )

                # Stop if centroids have minimal movement

                if max\_centroid\_shift < 0.0001:

                    break

                # Update centroids

                self.centroids = new\_centroids

* calculate\_distance(...): Tính khoảng cách Euclid giữa các tâm cụm cũ và mới.
* max\_centroid\_shift: Giá trị thay đổi lớn nhất giữa các tâm cụm.
* **Dừng vòng lặp**: Nếu thay đổi nhỏ hơn 0.0001, coi như hội tụ. Nếu không: cập nhật self.centroids với các tâm cụm mới.

def transform(self, dataframe, feature\_columns):

        df = dataframe.withColumn('features', F.array(feature\_columns))

        assign\_cluster\_udf = F.udf(self.find\_closest\_cluster, IntegerType())

        return df.withColumn(self.predictCol,assign\_cluster\_udf(F.col('features')))

* Hàm này áp dụng model K-Means đã huấn luyện để dự đoán cụm cho tập dữ liệu mới.
* **Tạo cột** features: Tương tự như hàm fit, gộp các cột đặc trưng (feature\_columns) thành một mảng.
* **Tạo cột** features: Tương tự như hàm fit, gộp các cột đặc trưng (feature\_columns) thành một mảng.
* **Thêm cột dự đoán**: Tạo cột mới (self.predictCol, mặc định là prediction) chứa chỉ số của cụm mà mỗi điểm được gán vào.

## DBScan

### Cơ sở lý thuyết

* DBSCAN (Density-Based Spatial Clustering of Applications with Noise) là một thuật toán nổi bật được giới thiệu bởi Ester, Kriegel, Sander, và Xu vào năm 1996. DBSCAN dựa trên khái niệm “phân cụm dựa trên mật độ”. Phương pháp này xác định cụm dữ liệu là các khu vực có mật độ cao được tách biệt bởi các khu vực có mật độ thấp. Thuật toán đặc biệt có khả năng xử lý nhiễu và phát hiện cụm dữ liệu với hình dạng phức tạp.
* Các định nghĩa: [1] [2] [3] [4]
  + Eps-neighborhood: vùng lân cận của một điểm dữ liệu P là tập hợp tất cả các điểm dữ liệu nằm trong phạm vi bán kính epsilon (e) xung quanh điểm P. Kí hiệu của tập hợp điểm này là:

* + - Trong đó:
      * D là tập hợp tất cả các điểm dữ liệu của tập huấn luyện.
      * là khoảng cách giữa điểm P và Q.

A diagram of circles with arrows and dots

Description automatically generated

Figure 54. Eps-neighborhood

* + Khả năng tiếp cận trực tiếp mật độ(directly density-reachable): dùng để xác định xem một điểm dữ liệu có thể được kết nối với các điểm dữ liệu nằm trong vùng lân cận epsilon hay không. Điểm P được coi là có thể tiếp cận trực tiếp tới điểm Q(cùng với tham số epsilon và minPts) nếu thỏa mãn điều kiện sau:
    - Q nằm trong vùng lân cận epsilon:
    - Số lượng các điểm dữ liệu nằm trong vùng lân cận epsilon tối thiểu là minPts:

A diagram of circles and dots

Description automatically generated

Figure 55. Directly density-reachable

* + Khả năng tiếp cận mật độ(density-reachable): liên quan đến hình thành một chuỗi liên kết các điểm trong cụm. Điểm p có khả năng tiếp cận mật độ bởi điểm q nếu dựa trên tham số eps và minPts, có một chuỗi các điểm với và mà có khả năng tiếp cận trực tiếp mật độ (directly density-reachable) từ .

A diagram of circles and dots

Description automatically generated

Figure 56. Density-reachable

* + Kết nối mật độ(density-connected): một điểm p là kết nối mật độ(density-connected) đến 1 điểm q khi và chỉ khi có một điểm o mà cả hai điểm p và q đều có khả năng tiếp cận mật độ(density-reachable) từ o. Một cụm là một tập các điểm density-connected lớn nhất. Nhiễu là điểm không thuộc cụm nào.

A diagram of circles and dots

Description automatically generated

Figure 57. Density-connected

* + Cụm(cluster): Gọi là một tập con không rỗng của tập dữ liệu D thỏa mãn:
    - và q có khả năng tiếp cận mật độ(density-reachable) từ p thì .
    - p kết nối mật độ(density-connected) từ q.
  + Điểm lõi (core point): một điểm được coi là điểm lõi nếu ít nhất minPts điểm nằm trong bán kính epsilon.
  + Điểm biên (border point): các điểm không phải là điểm lõi nhưng nằm trong bán kính epsilon của một điểm lõi.
  + Điểm nhiễu (noise point): các điểm không phải là điểm lõi hoặc điểm biên.

A diagram of circles and lines

Description automatically generated

Figure 58. Phân loại các điểm trong DBScan

* + Tham số:
    - epsilon ( > 0): bán kính lớn nhất vùng lân cận.
    - minPts(ngưỡng): số lượng tối thiểu điểm lân cận (bao gồm chính nó) cần thiết để một điểm trở thành core point (điểm lõi).

1.  DBSCAN(DB, distFunc, eps, minPts) {

C := 0 // Bộ đếm cụm

for each point P in database DB {

if label(P) ≠ undefined then continue // Đã được xử lý trước đó

Neighbors N := RangeQuery(DB, distFunc, P, eps) // Tìm hàng xóm

if |N| < minPts then {

label(P) := Noise // Gán nhãn là Nhiễu

continue

}

C := C + 1 // Nhãn cụm tiếp theo

label(P) := C // Gán nhãn cho điểm ban đầu

ExpansionSet S := N \ {P} // Tập hợp các điểm lân cận để mở rộng

for each point Q in S {

if label(Q) = Noise then label(Q) := C // Thay đổi Nhiễu thành điểm biên

if label(Q) ≠ undefined then continue // Đã được xử lý trước đó

label(Q) := C // Gán nhãn cho hàng xóm

Neighbors N := RangeQuery(DB, distFunc, Q, eps) // Tìm hàng xóm

if |N| ≥ minPts then {

ExpansionSet := ExpansionSet ∪ N // Thêm hàng xóm mới vào tập hợp mở rộng

}

}

}

}

### Thực nghiệm

* Các bước thực hiện:
  + Bước 1: Xử lý dữ liệu đầu vào: Chọn những cột cần tính toán, loại bỏ các dòng có giá trị null, đánh số thứ tự.
  + Bước 2: Tính khoảng cách Euclidean giữa các điểm dữ liệu, bao gồm chính điểm đó.
  + Bước 3: Khai báo bán kính lân cận eps và số điểm lân cận nhỏ nhất minPts, broadcast hai giá trị này.
  + Bước 4: Tìm hàng xóm cho tất cả các điểm. Hàng xóm của một điểm cũng bao gồm chính điểm đó.
  + Bước 5: Phân loại các điểm core, border, noise dựa trên số lượng hàng xóm và minPts, khởi tạo cluster id ban đầu cho các điểm là 0.
  + Bước 6: Khởi tạo 1 biến cluster\_id = 0. Chọn một điểm core từ Dataframe. Lấy các điểm hàng xóm từ điểm core đã chọn, tăng biến cluster\_id lên 1, cập nhật cluster\_id cho các điểm này bao gồm điểm core đã chọn.
  + Bước 7: Từ điểm core point, mở rộng cụm dựa vào hàng xóm của các điểm đã cập nhật cluster id ở trên. Chỉ thêm các điểm core và border vào cụm. Dừng khi không tìm thấy điểm core và border trong các hàng xóm.
  + Bước 8: Lặp lại bước 6 đến khi không tìm thấy điểm core trong Dataframe.

A screenshot of a computer

Description automatically generated

Figure 59. Mô tả song song hóa giải thuật DBScan

* Chạy thuật toán:
  + Bước 1: Xử lý dữ liệu đầu vào: Chọn những cột cần tính toán, loại bỏ các dòng có giá trị null, đánh số thứ tự.

A screenshot of a computer

Description automatically generated

Figure 60. Xử lý dữ liệu đầu vào

* + Bước 2: Tính khoảng cách Euclidean giữa các điểm dữ liệu, bao gồm chính điểm đó.

A screen shot of a computer

Description automatically generated

Figure 61. Tính khoảng cach Euclidean giữa các điểm

* + Bước 3: Khai báo bán kính lân cận eps và số điểm lân cận nhỏ nhất minPts, broadcast 2 giá trị này.

A computer screen shot of a computer

Description automatically generated

Figure 62. Khởi tạo eps và minPts và broadcast giá trị

* + Bước 4: Tìm hàng xóm cho tất cả các điểm. Hàng xóm của một điểm cũng bao gồm chính điểm đó.

A computer screen shot of a computer code

Description automatically generated

Figure 63. Tìm hàng xóm cho tất cả các điểm

* + Bước 5: Phân loại các điểm core, border, noise dựa trên số lượng hàng xóm và minPts, khởi tạo cluster id ban đầu cho các điểm là 0.

A screen shot of a computer

Description automatically generated

Figure 64. Phân loại điểm core và khởi tạo cluster ban đầu cho các điểm

A close-up of a computer screen

Description automatically generated

Figure 65. Tìm các điểm noise

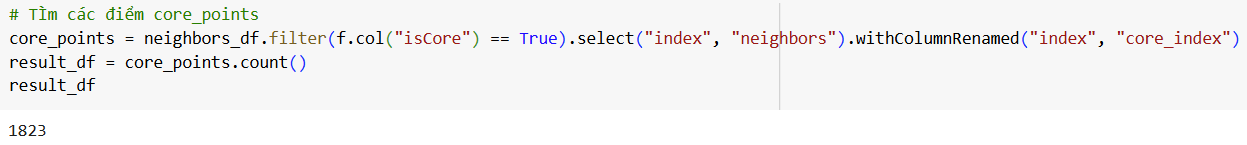


Figure 66. Tìm các điểm core

A screen shot of a computer

Description automatically generated

Figure 67. Tìm các điểm border

A screen shot of a computer code

Description automatically generated

Figure 68. Cập nhật các điểm border vào dataframe

* + Bước 6: Khởi tạo 1 biến cluster\_id = 0. Chọn một điểm core từ Dataframe. Lấy các điểm hàng xóm từ điểm core đã chọn, tăng biến cluster\_id lên 1, cập nhật cluster\_id cho các điểm này bao gồm điểm core đã chọn.

A screenshot of a computer code

Description automatically generated

Figure 69. Mở rộng cụm từ một điểm core

* + Bước 7: Từ điểm core point, mở rộng cụm dựa vào hàng xóm của các điểm đã cập nhật cluster\_id ở trên. Chỉ thêm các điểm core và border vào cụm. Dừng khi không tìm thấy điểm core và border trong các hàng xóm.

A screenshot of a computer program

Description automatically generated

Figure 70. Hàm mở rộng cụm

* + Bước 8: Lặp lại bước 6 đến khi không tìm thấy điểm core trong Dataframe.

# KẾT QUẢ ĐẠT ĐƯỢC

## Kết quả

### K-Means

A screen shot of a computer

Description automatically generated

Figure 71 Số lượng dữ liệu ở mỗi cụm

* Nhận xét:
  + Dữ liệu có sự chênh lệch lớn giữa 2 cụm. Cụm lớn chứa đa số dữ liệu trong khi cụm nhỏ chỉ chiếm một phần rất nhỏ (khoảng 3%)
  + Cụm lớn:
    - Chứa phần lớn dữ liệu, có thể đại diện cho nhóm chính trong tập dữ liệu.
    - Các điểm trong cụm này có thể có nhiều đặc điểm tương đồng dẫn đến việc gom vào cùng một cụm.
  + Cụm nhỏ:
    - Có thể chứa các **dữ liệu ngoại lệ** hoặc nhóm dữ liệu đặc biệt với các đặc trưng khác biệt rõ rệt so với cụm lớn.

### DBScan

A screenshot of a computer code

Description automatically generated

Figure 72. Kết quả phân cụm bằng DBScan

* Nhận xét:
* Cụm ID = 1:
* Đặc điểm: Đây là cụm lớn nhất, chiếm tới 97.76% tổng dữ liệu.
* Ý nghĩa: Cụm này có thể đại diện cho các sản phẩm phổ biến và tiêu chuẩn trên Amazon. Các sản phẩm trong cụm này có thông số kỹ thuật tương tự nhau, thể hiện tính đồng nhất và phổ biến của các mặt hàng.
* Đây là nhóm sản phẩm trọng tâm và mang tính chính thống, phản ánh phần lớn thị trường.
* Cụm ID = 2:
* Đặc điểm: Cụm này nhỏ hơn nhiều so với ID = 1, chỉ chiếm 1.97% tổng dữ liệu
* Ý nghĩa: Cụm này có thể đại diện cho các sản phẩm có thông số kỹ thuật đặc biệt hoặc khác biệt so với cụm chính. Có thể đây là các sản phẩm ngách, cao cấp, hoặc chuyên biệt, ít phổ biến hơn nhưng có đặc tính nổi trội.
* Cho thấy sự hiếm hoi hoặc độc đáo của các sản phẩm này trên thị trường
* Cụm ID = 0:
* Đặc điểm: Đây là các điểm nhiễu, chưa phân cụm, chiếm một phần rất nhỏ của dữ liệu (0.27%)
* Đây là các sản phẩm ngoại lệ hoặc dữ liệu sai lệch hoặc thiếu thông tin về thông số kỹ thuật.

## So sánh, đánh giá

### K-Means

* Sử dụng SilhouetteEvaluator tự cài đặt và thư viện máy học ClusteringEvaluator để đánh giá chất lượng cụm
  + Đầu tiên là chỉ số silhouette score bằng việc cài đặt không sử dụng thư viện máy học

for k in range(2,11):

kmean = K\_Mean(k)

kmean.fit(df, feature\_cols)

df\_fit = kmean.transform(df, feature\_cols)

evaluator = SilhouetteScore()

silhouette\_score = evaluator.evaluate(df\_fit)

silhouette\_list.append(silhouette\_score)

A graph of a number of clusters

Description automatically generated

Figure 73 Chỉ số silhouette score cài đặt không dùng thư viện máy học

* + Tiếp theo là chỉ số silhouette score với việc sử dụng thư viện máy học

for k in range(2,11):

    kmeans = KMeans(k=k, seed=1)

    kmeans = kmeans.setFeaturesCol("features")

    model = kmeans.fit(vectorized\_df)

    df\_result = model.transform(vectorized\_df)

    evaluator = ClusteringEvaluator()

    silhouette\_score = evaluator.evaluate(df\_result)

    silhouette\_list.append(silhouette\_score)

A graph of a number of clusters

Description automatically generated

Figure 74 Chỉ số silhouette score cài đặt dùng thư viện máy học

* So sánh việc phân cụm của việc tự cài đặt và thư viện máy học
  + Dựa vào silhouette score, ta chọn cụm k=2

A screenshot of a computer

Description automatically generated

Figure 75 Phân cụm không dùng thư viện máy học

A computer screen shot of a program

Description automatically generated

Figure 76 Phân cụm sử dụng thư viện máy học

* Nhận xét:
  + Silhouette score của cả 2 cách cài đặt từ k=2 tới k=7 khá là xêm xêm nhau, nhưng từ k=8 tới k=10 thì lệch đi khá nhiều. Điều đó cho thấy cách sử dụng thư viện làm tốt hơn.
  + Bên cạnh đó tốc độ thực thi khi sử dụng thư viện máy học nhanh hơn việc tự cài đặt rất nhiều

### DBScan

Silhouette Score = 0.855



Figure 77. Chỉ số Silhoutte của DBScan

* Nhận xét:
* Kết quả cho thấy DBSCAN hoạt động rất hiệu quả trên tập dữ liệu này.
* Các cụm được phân tách tốt, có độ gắn kết cao và ít bị ảnh hưởng bởi điểm nhiễu.
* DBSCAN phù hợp với cấu trúc dữ liệu hiện tại, chứng tỏ được khả năng mạnh mẽ trong việc phân cụm dữ liệu mật độ cao và có nhiễu.

Heatmap

A graph of a number of clusters

Description automatically generated

Figure 78. Heatmap của DBScan

* Nhận xét:
* Ảnh hưởng của epsilon (eps):
* Khi eps = 0.15, số lượng cụm là 3 (cho cả minPts = 10 và 12).
* Khi eps tăng lên 0.2, số lượng cụm giảm xuống còn 2.
* Điều này cho thấy giá trị eps nhỏ hơn giúp phát hiện nhiều cụm hơn, còn eps lớn làm các cụm bị gộp lại.
* Ảnh hưởng của minPts: Thay đổi minPts từ 10 lên 12 không làm thay đổi số lượng cụm trong cả hai trường hợp eps = 0.15 và eps = 0.2.
* Điều này có nghĩa minPts trong khoảng này chưa ảnh hưởng đáng kể đến kết quả phân cụm.

# KẾT LUẬN

## Ưu điểm

* Nhận diện được bài toán khai thác dữ liệu lớn và áp dụng được các giải thuật khai thác dữ liệu song song, phân tán.
* Ứng dụng công nghệ học máy hiện đại: Đề tài sử dụng các thuật toán phân cụm tiên tiến như K-means và DBScan đại diện cho ứng dụng thực tiễn của học máy trong việc phân tích dữ liệu của sản phẩm.

## Hạn chế

* Bộ dữ liệu crawl trực tiếp từ trang web của Amazon, nên việc dữ liệu có thể bị thiếu các thuộc tính là không thể tránh khỏi, dẫn tới việc tiền xử lý gặp nhiều khó khăn.
* Nhóm chọn K-means để phân tích dữ liệu, nó phụ thuộc vào việc xác định số lượng cụm k ban đầu và cách xử lý dữ liệu đầu vào, nên còn môt số hạn chế về việc cải thiện hiệu suất của thuật toán. Còn về DBScan, thuật toán này cũng yêu cầu 2 tham số là eps và minPts, việc chọn giá trị cho các tham số này không phải lúc nào cũng dễ dàng, kết quả phân cụm có thể không chính xác hoặc không hợp lý. Việc tìm giá trị tối ưu cho các tham số này phụ thuộc vào dữ liệu, và trong một số trường hợp, điều này có thể mất rất nhiều thời gian và công sức.
* Cả 2 thuật toán đều có thể không chính xác nếu dữ liệu có nhiều nhiễu hoặc nếu các cụm không rõ ràng.

## Hướng phát triển

* Thực hiện crawl dữ liệu sản phẩm trên nhiều sàn thương mại điện tử hơn như Tiki, Lazada, Alibaba,…. Để có nhiều dữ liệu để phân tích cũng như huấn luyện mô hình.
* Tối ưu hoá hiệu suất, cải thiện tốc độ xử lý của 2 thuật toán K-Means và DBScan.
* Nghiên cứu và áp dụng xử lý dữ liệu theo thời gian thực, đưa ra kết quả dự đoán real-time, hỗ trợ trong mục đính, nhiều đề tài, nhiều chủ đề khác không chỉ dừng ở các sản phẩm được bày bán trên các trang thương mại điện tử.

# PHÂN CÔNG CÔNG VIỆC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Nguyễn Hoàng Đăng Khoa (21520999) | Cù Ngọc Hoàng (21522086) | Nguyễn Trần Gia Kiệt (21522258) (Nhóm trưởng) | Bùi Đình Triệu (21521576) |
| Tổng quan đề tài | **x** | **x** | **x** | x |
| Lấy dữ liệu từ Amazon |  |  |  | x |
| Kỹ thuật tiền xử lý | **x** |  |  |  |
| Kỹ thuật khai thác dữ liệu |  |  |  |  |
| K-Means |  |  | **x** |  |
| DBScan |  | **x** |  |  |
| Kết quả đạt được |  |  |  |  |
| K-Means |  |  | **x** |  |
| DBScan |  | **x** |  |  |
| Kết luận |  |  | **x** |  |
| Làm báo cáo | **x** | **x** | **x** | x |
| Làm slide | **x** | **x** | **x** | x |
| Hoàn thành (%) | 100% | 100% | 100% | 100% |

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | P. D. Khanh, “Phương pháp phân cụm dựa trên mật độ (Density-Based Clustering),” Deep AI KhanhBlog, 2021. [Trực tuyến]. Available: https://phamdinhkhanh.github.io/deepai-book/ch\_ml/DBSCAN.html. |
| [2] | Sachinsoni, “Clustering Like a Pro: A Beginner’s Guide to DBSCAN,” Medium, 2023 December 2023. [Trực tuyến]. |
| [3] | D. P. Long, *Tài liệu bài giảng: KHAI THÁC DỮ LIỆU – IS252 - Chương 7: Gom cụm,* Ho Chi Minh: UIT. |
| [4] | O. Yenigün, “DBSCAN Clustering Algorithm Demystified,” builtin, 11 March 2024. [Trực tuyến]. Available: https://builtin.com/articles/dbscan. |
| [5] | Geeksforgeeks, “K means Clustering – Introduction,” [Trực tuyến]. Available: https://www.geeksforgeeks.org/k-means-clustering-introduction/. |
| [6] | Geeksforgeek, “K-Means Clustering using PySpark Python,” [Trực tuyến]. Available: https://www.geeksforgeeks.org/k-means-clustering-using-pyspark-python/. |
| [7] | E. Ding, “From Scratch: How to Code K-Means in Python (No Sklearn) for Machine Learning Interviews!,” [Trực tuyến]. Available: https://www.youtube.com/watch?v=uLs-EYUpGAw. |
| [8] | A. Tripathi, “Kmeans from Scratch with Silhoutte and elbow curve,” Kaggle, [Trực tuyến]. Available: https://www.kaggle.com/code/achintyatripathi/kmeans-from-scratch-with-silhoutte-and-elbow-curve. |
| [9] | D. P. Long, *Tài liệu bài giảng: KHAI THÁC DỮ LIỆU - IS252 - Chương 2: Tiền xử lý dữ liệu,* Ho Chi Minh: UIT. |
| [10] | geeksforgeeks, “Data Preprocessing in Data Mining,” [Trực tuyến]. Available: https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/. |
| [11] | Airflow, “Documentation” [Trực tuyến]. Available: https://airflow.apache.org/docs/. |
| [12] | Kmeans, “k-means clustering” [Trực tuyến]. Available: https://en.wikipedia.org/wiki/K-means\_clustering. |